Internet
11/26/03

all considered

THE HISTORY CHANNEL
REMEMBER THE ALAMO
Dec. 16 9PM/8C
ENTER TO WIN a Grand Prize Trip to Mexico & San Antonio, Texas!
Clic

Homepage | Advanced Search

Search using: HotBot | Lycos | Google | Ask Jeeves

CUSTOM WEB FILTERS

Tools | HotBot Skins | Preferences

Date: Before October 18 1999 [Edit this Search]

WEB RESULTS (Showing Results 1 - 50 of 2,794)

1. Autoconf
Creating Automatic Configuration Scripts Edition 2.12, for Autoconf version 2.12 by David MacKenzie Table of Contents A physicist, an engineer, and a computer scientist were discussing the nature of...
www.primate.wisc.edu/software/autoconf/autoconf.html - July 14, 1997 - 267 KB

2. RPI FRISC Group Software Configuration
FRISC Group - Software Configuration and Problems Standard paper Sizes A0 841x1149mm A1 594x841mm A2 420x594mm A3 297x420mm A4 210x297mm Color Transparencies and Hardcopies on RCS printers (Atul - 3/10/94) 1. ... RS6000 login configuration. Automatic Propagation of DISPLAY variable ... problem (declaring fixed fonts). eXodus configuration on Suns ...
inp.cie.rpi.edu/research/mcdonald/frisc/docs/software_config.html - March 31, 1997 - 248 KB

3. PAUL'S MCSE PAGE
The Windows 95 Exam 70-63 Implementing and Supporting Microsoft Windows 95 70 Questions, 90 Minutes (1.28 Minutes per question) Passing Score 714/1000 points (14.28 ea.) Planning & Installation Architecture and Memory Dial-up Networking 32-bit and
website.lineone.net/~paulcon/winexam.htm - May 22, 1999 - 212 KB

4. Configuration Challenges in a Global Cache Hierarchy
Evolution of the NLANR Cache Hierarchy: Global Configuration Challenges Duane Wessels and k claffy UC, San Diego World-Wide Web caches are designed to alleviate some of the problems imposed by ever-increasing Internet traffic growth.
www.nlanr.net/Papers/Cache96 - November 10, 1996 - 46 KB

5. OpenVMS VAX Version 6.2 Upgrade and Installation Manual
Â© Digital Equipment Corporation 1995. All rights reserved. OpenVMS VAX Version 6.2 Upgrade and Installation Manual This manual provides step-by-step instructions for upgrading and installing the OpenVMS VAX operating system.
www.ibb.net/~telkamp/vax/vms-inst.htm - August 20, 1996 - 533 KB

6. Inside Windows 98 -- Ch 3 -- Advanced Setup
... required to determine an optimal or preferred configuration for the ... make it easier to upgrade

components or drivers from ... a shared configuration it is easier to upgrade and maintain ......
cma.zdnet.com/book/insidewin98/ch03/ch03.htm - December 8, 1998 - 32 KB

7. the case for IPv6
Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. ... scalability, security, ease-of-configuration, and network ... not need operators (automatic or otherwise) to ... configuration tool because it maintains static tables that determine which ...
hegel.ittc.ukans.edu/topics/internet/internet-drafts/draft-i/draft-ietf-iab - April 7, 1998 - 111 KB

8. Toward Automatic Generation of Novice User Test Scripts
... Toward Automatic Generation of Novice User Test Scripts ... find program failures rather than to determine usability characteristics. ... difficult to manage and upgrade to the next version. ......
www.acm.org/sigchi/chi96/proceedings/papers/Kasik/djk_txt.htm - October 4, 1996 - 43 KB

9. Windows 98 Installation & Configuration Handbook -- Ch 28 --...
... Windows 98 Installation & Configuration Handbook. - 28 ... can specify the button configuration for a right- or left-handed ... If Windows can determine the make and model of the ...
cma.zdnet.com/book/win98config/ch28/ch28.htm - October 15, 1998 - 30 KB

10. nia-37
... to determine which ... Determine the types of connections that will be made to the network: Ethernet, synchronous DDCMP, or asynchronous DDCMP connections. You can use the network configuration...
www.etext.org/CuD/NIA/nia-37 - December 20, 1990 - 102 KB

11. Automation of Site Configuration Management
Automation of Site Configuration Management Jon Finke - Rensselaer Polytechnic Institute ... track and manage the configuration of large numbers of Unix ... week after the successful upgrade of two...
www.rpi.edu/~finkej/Papers/LISA97-SiteConf.html - November 6, 1997 - 57 KB

12. NCKCN Mac CD Installation Guide
... to determine your system software version and if you have Open Transport installed. If you need to upgrade and install ... it is set to Automatic Configuration. We will ...
www.nckcn.com/NCKCN/mac/MacCD.htm - July 14, 1999 - 27 KB

13. Before You Begin
... FrameSaver 9620. R1-to-R2 Upgrade. Installation Instructions ... DBM for backup and automatic DLCI configuration and cross-connection. ... get a file listing so you can determine the latest ......
www.paradyne.com/technical_manuals/9621-A2-GN11-10.pdf - February 5, 1999 - 37 KB

14. untitled
... CONFIGURATION MANAGEMENT PROGRAM PLAN ... The TWRS Configuration Management Program Plan ... upgrade, reconstitute, and maintain consistency among the requirements, product configuration, ...
www.hanford.gov/twrs/cmpp/twrscmpp.htm - February 19, 1999 - 98 KB

15. Autoconf
Creating Automatic Configuration Scripts Edition 2.12, for Autoconf version 2.12 by David MacKenzie A physicist, an engineer, and a computer scientist were discussing the nature of God. ... The configuration scripts produced by Autoconf require no ... Autoconf macro files, to determine which macros to output ...
www.amath.washington.edu/~lf/tutorials/autoconf/autoconf/autoconf.html - March 5, 1998 - 260 KB

Autoconf

Creating Automatic Configuration Scripts

Edition 2.12, for Autoconf version 2.12

November 1996

by David MacKenzie

Introduction

```
A physicist, an engineer, and a computer scientist were
discussing the nature of God.  Surely a Physicist, said the
physicist, because early in the Creation, God made Light; and you
know, Maxwell's equations, the dual nature of electro-magnetic
waves, the relativist consequences... An Engineer!, said the
engineer, because before making Light, God split the Chaos into
Land and Water; it takes a hell of an engineer to handle that big
amount of mud, and orderly separation of solids from
liquids... The computer scientist shouted: And the Chaos,
where do you think it was coming from, hmm?

---Anonymous
```

Autoconf is a tool for producing shell scripts that automatically configure software source code packages to adapt to many kinds of UNIX-like systems. The configuration scripts produced by Autoconf are independent of Autoconf when they are run, so their users do not need to have Autoconf.

The configuration scripts produced by Autoconf require no manual user intervention when run; they do not normally even need an argument specifying the system type. Instead, they test for the presence of each feature that the software package they are for might need individually. (Before each check, they print a one-line message stating what they are checking for, so the user doesn't get too bored while waiting for the script to finish.) As a result, they deal well with systems that are hybrids or customized from the more common UNIX variants. There is no need to maintain files that list the features supported by each release of each variant of UNIX.

For each software package that Autoconf is used with, it creates a configuration script from a template file that lists the system features that the package needs or can use. After the shell code to recognize and respond to a system feature has been written, Autoconf allows it to be shared by many software packages that can use (or need) that feature. If it later turns out that the shell code needs adjustment for some reason, it needs to be changed in only one place; all of the configuration scripts can be regenerated automatically to take advantage of the updated code.

The Metaconfig package is similar in purpose to Autoconf, but the scripts it produces require manual user intervention, which is quite inconvenient when configuring large source trees. Unlike Metaconfig scripts, Autoconf scripts can support cross-compiling, if some care is taken in writing them.

There are several jobs related to making portable software packages that Autoconf currently does not do. Among these are automatically creating `` `Makefile' `` files with all of the standard targets, and supplying replacements for standard library functions and header files on systems that lack them. Work is in progress to add those features in the future.

Autoconf imposes some restrictions on the names of macros used with `#ifdef` in C programs (see section Preprocessor Symbol Index).

Autoconf requires GNU `m4` in order to generate the scripts. It uses features that some UNIX versions of `m4` do not have. It also overflows internal limits of some versions of `m4`, including GNU `m4` 1.0. You must use version 1.1 or later of GNU `m4`. Using version 1.3 or later will be much faster than 1.1 or 1.2.

See section Upgrading From Version 1, for information about upgrading from version 1. See section History of Autoconf, for the story of Autoconf's development. See section Questions About Autoconf, for answers to some common questions about Autoconf.

Mail suggestions and bug reports for Autoconf to `bug-gnu-utils@prep.ai.mit.edu`. Please include the Autoconf version number, which you can get by running `` `autoconf --version' ``.

Making `configure` Scripts

The configuration scripts that Autoconf produces are by convention called `configure`. When run, `configure` creates several files, replacing configuration parameters in them with appropriate values. The files that `configure` creates are:

- one or more `` `Makefile' `` files, one in each subdirectory of the package (see section Substitutions in Makefiles);
- optionally, a C header file, the name of which is configurable, containing `#define` directives (see section Configuration Header Files);
- a shell script called `` `config.status' `` that, when run, will recreate the files listed above (see section Recreating a Configuration);
- a shell script called `` `config.cache' `` that saves the results of running many of the tests (see section Cache Files);

- a file called `config.log` containing any messages produced by compilers, to help debugging if `configure` makes a mistake.

To create a `configure` script with Autoconf, you need to write an Autoconf input file `configure.in` and run `autoconf` on it. If you write your own feature tests to supplement those that come with Autoconf, you might also write files called `aclocal.m4` and `acsite.m4`. If you use a C header file to contain `#define` directives, you might also write `acconfig.h`, and you will distribute the Autoconf-generated file `config.h.in` with the package.

Here is a diagram showing how the files that can be used in configuration are produced. Programs that are executed are suffixed by `*`. Optional files are enclosed in square brackets (`[]`). `autoconf` and `autoheader` also read the installed Autoconf macro files (by reading `autoconf.m4`).

Files used in preparing a software package for distribution:

```
your source files --> [autoscan*] --> [configure.scan] --> configure.in

configure.in --.   .------> autoconf* -----> configure
               +---+
[aclocal.m4] --+   `---.
[acsite.m4] ---'       |
                       +--> [autoheader*] -> [config.h.in]
[acconfig.h] ----.     |
                 +-----'
[config.h.top] --+
[config.h.bot] --'

Makefile.in -------------------------------> Makefile.in
```

Files used in configuring a software package:

```
                       .-------------> config.cache
configure* ------------+-------------> config.log
                       |
[config.h.in] -.       v            .-> [config.h] -.
               +--> config.status* -+               +--> make*
Makefile.in ---'                    `-> Makefile ---'
```

Writing `configure.in`

To produce a `configure` script for a software package, create a file called `configure.in` that contains invocations of the Autoconf macros that test the system features your package needs or can use. Autoconf macros already exist to check for many features; see section Existing Tests, for their descriptions. For most other features, you can use Autoconf template macros to produce custom checks; see section Writing Tests, for information about them. For especially tricky or specialized features, `configure.in` might need to contain some hand-crafted shell commands. The `autoscan` program can give you a good start in writing `configure.in` (see section Using `autoscan` to Create `configure.in`, for more information).

The order in which `` `configure.in' `` calls the Autoconf macros is not important, with a few exceptions. Every `` `configure.in' `` must contain a call to `AC_INIT` before the checks, and a call to `AC_OUTPUT` at the end (see section Creating Output Files). Additionally, some macros rely on other macros having been called first, because they check previously set values of some variables to decide what to do. These macros are noted in the individual descriptions (see section Existing Tests), and they also warn you when creating `configure` if they are called out of order.

To encourage consistency, here is a suggested order for calling the Autoconf macros. Generally speaking, the things near the end of this list could depend on things earlier in it. For example, library functions could be affected by typedefs and libraries.

```
AC_INIT(file)
checks for programs
checks for libraries
checks for header files
checks for typedefs
checks for structures
checks for compiler characteristics
checks for library functions
checks for system services
AC_OUTPUT([file...])
```

It is best to put each macro call on its own line in `` `configure.in' ``. Most of the macros don't add extra newlines; they rely on the newline after the macro call to terminate the commands. This approach makes the generated `configure` script a little easier to read by not inserting lots of blank lines. It is generally safe to set shell variables on the same line as a macro call, because the shell allows assignments without intervening newlines.

When calling macros that take arguments, there must not be any blank space between the macro name and the open parenthesis. Arguments can be more than one line long if they are enclosed within the `m4` quote characters `` `[' `` and `` `]' ``. If you have a long line such as a list of file names, you can generally use a backslash at the end of a line to continue it logically on the next line (this is implemented by the shell, not by anything special that Autoconf does).

Some macros handle two cases: what to do if the given condition is met, and what to do if the condition is not met. In some places you might want to do something if a condition is true but do nothing if it's false, or vice versa. To omit the true case, pass an empty value for the *action-if-found* argument to the macro. To omit the false case, omit the *action-if-not-found* argument to the macro, including the comma before it.

You can include comments in `` `configure.in' `` files by starting them with the `m4` builtin macro `dnl`, which discards text up through the next newline. These comments do not appear in the generated `configure` scripts. For example, it is helpful to begin `` `configure.in' `` files with a line like this:

```
dnl Process this file with autoconf to produce a configure script.
```

Using `autoscan` to Create `configure.in'

The `autoscan` program can help you create a `configure.in' file for a software package. `autoscan` examines source files in the directory tree rooted at a directory given as a command line argument, or the current directory if none is given. It searches the source files for common portability problems and creates a file `configure.scan' which is a preliminary `configure.in' for that package.

You should manually examine `configure.scan' before renaming it to `configure.in'; it will probably need some adjustments. Occasionally `autoscan` outputs a macro in the wrong order relative to another macro, so that `autoconf` produces a warning; you need to move such macros manually. Also, if you want the package to use a configuration header file, you must add a call to `AC_CONFIG_HEADER` (see section Configuration Header Files). You might also have to change or add some `#if` directives to your program in order to make it work with Autoconf (see section Using `ifnames` to List Conditionals, for information about a program that can help with that job).

`autoscan` uses several data files, which are installed along with the distributed Autoconf macro files, to determine which macros to output when it finds particular symbols in a package's source files. These files all have the same format. Each line consists of a symbol, whitespace, and the Autoconf macro to output if that symbol is encountered. Lines starting with `#' are comments.

`autoscan` is only installed if you already have Perl installed. `autoscan` accepts the following options:

`--help`
: Print a summary of the command line options and exit.

`--macrodir=`*`dir`*
: Look for the data files in directory *dir* instead of the default installation directory. You can also set the `AC_MACRODIR` environment variable to a directory; this option overrides the environment variable.

`--verbose`
: Print the names of the files it examines and the potentially interesting symbols it finds in them. This output can be voluminous.

`--version`
: Print the version number of Autoconf and exit.

Using `ifnames` to List Conditionals

`ifnames` can help when writing a `configure.in' for a software package. It prints the identifiers that the package already uses in C preprocessor conditionals. If a

---

This document was generated on 5 March 1998 using the texi2html translator version 1.52.